Pages 1 - 90

UNITED STATES DISTRICT COURT

NORTHERN DISTRICT OF CALIFORNIA

Before The Honorable Beth Labson Freeman, Judge

Cisco Systems, Inc.,)
)
Plaintiff,)
)
VS.) **NO. C 14-5344**
)
Arista Networks, Inc.,)
)
Defendant.)
)

San Francisco, California
Friday, March 11, 2016

TRANSCRIPT OF PROCEEDINGS OF THE OFFICIAL ELECTRONIC SOUND RECORDING

FTR: 1:32 p.m. to 3:40 p.m.

APPEARANCES:

For Plaintiff:

Quinn Emanuel Urquhart & Sullivan, LLP
50 California -- Floor 22
Redwood City, California 94063
BY: SEAN SANG-CHUL PAK, ESQUIRE

For Defendant:

Keker & Van Nest LLP
633 Battery Street
San Francisco, CA 94111
BY: AJAY KRISHNAN, ESQUIRE
DAVID SILBERT, ESQUIRE
DAVID ROSEN, ESQUIRE

Transcribed By: Pamela A. Batalo,
Transcriber

Friday, March 11, 2016 1:32 p.m.

P-R-O-C-E-E-D-I-N-G-S

---000---

MR. PAK: We have today (inaudible) as well as Dr. Allan Rose, who will be observing but not presenting today.

THE COURT: Excellent. Thank you.

MR. SILBERT: Good afternoon, Your Honor. David Silvert of Keker & Van Nest on behalf of the defendant Arista.

MR. KRISHNAN: Good afternoon, Your Honor. Ajay Krishnan from Keker & Van Nest.

THE COURT: Hello.

MR. ROSEN: David Rosen from Keker & Van Nest.

THE COURT: Welcome. All right. This is our time for tutorial. And I want to get down to -- and I guess I may as well tell you the bad news that -- well, actually I left my calendar at my desk. I'm going to have to postpone the hearing next week. As you can see, I'm in the middle of a little small trial here. And it's -- well, I just need the day for trial. So -- but I don't have to postpone it much. I mean, it's really just a matter of working it in the next couple of weeks, and I also was interested, though, I know that IPR has been filed by Arista, and I believe you told me that the dates are two dates in May that you're expecting a response.

UNIDENTIFIED SPEAKER: That is correct.

THE COURT: So of course I didn't stay the case

because it's only pending IPR, but is it your desire just to go forward and have me construe the claims and then see what happens?

UNIDENTIFIED SPEAKER: Well, I -- personally, we don't think that that would make the most sense. Now, if -- there may be some challenges fitting everything into the schedule, but if the -- if the PTAB institutes, then obviously that's something Your Honor would want to take into account.

THE COURT: I do, yes. I'm not asking to postpone the claims construction hearing. It's really more about the timing of an order.

UNIDENTIFIED SPEAKER: Your Honor, we have a very tight schedule --

THE COURT: Yes, we do.

UNIDENTIFIED SPEAKER: And we have expert discovery that will take place immediately after patent (inaudible) and as Your Honor knows --

THE COURT: Yeah.

UNIDENTIFIED SPEAKER: -- we have a November trial date. So we are filing oppositions to these petitions --

THE COURT: Sure.

UNIDENTIFIED SPEAKER: -- on behalf of Cisco. We -- we believe that there is a substantial chance that the petitions will not be granted.

THE COURT: In which case I don't want to lose any

time.

UNIDENTIFIED SPEAKER: Absolutely, Your Honor. So I would suggest that if we move forward, we try to narrow the issues --

THE COURT: Okay.

UNIDENTIFIED SPEAKER: I think that hopefully after today's exciting presentation from both sides, that the issues will become clear --

THE COURT: Okay. All right. Well, we're certainly going to go forward, and as I say, I'm not available for the claims construction next Friday, but I'm -- I'm looking at in the next 10 days or so after that, so that I'm hoping it's just a little hiccup and not -- not meant to be a delay. But I've got a deadline with my jury and I like -- you know -- and when it's your trial, I'll be clearing the path for you as well.

All right. Well, we'll --

UNIDENTIFIED SPEAKER: One other (inaudible) that I would like to says. Actually, this relates to the antitrust case.

THE COURT: Oh.

UNIDENTIFIED SPEAKER: That's also now pending --

THE COURT: Of course that's not really on my radar much.

UNIDENTIFIED SPEAKER: Let me just tell you the issue and --

THE COURT: Okay.

UNIDENTIFIED SPEAKER: -- and you can say what you want to do with it.

But the Court set a case management conference in that case yesterday --

THE COURT: Okay.

UNIDENTIFIED SPEAKER: -- for may 26. Cisco, in its pleadings, has said that it intends to move to dismiss the Complaint. We have been talking to Cisco about an extension for them to file their brief probably sometime around mid April.

THE COURT: Oh, I see.

UNIDENTIFIED SPEAKER: Our only concern is knowing the congestion of the Court's calendar, if we wait until April to request a hearing date, that the hearing date is then going to be sometime significantly in the future --

THE COURT: You can expect it will be in July or August.

UNIDENTIFIED SPEAKER: Right. And so what we were hoping -- and I -- I understand the realities of the situation, but if it's possible to -- to reserve the date now, and -- for that motion and then perhaps Your Honor -- I don't know when the Court would first be available, but --

THE COURT: I'm not inclined to give the antitrust case any priority, other than its filing date priority. I

don't see that as being of the same significance as the copyright case, which I think is the lead issue in this case. And the patent -- because I don't think the patent infringement is as significant, but important -- don't get me wrong.

And I am glad to have the case management on the date that its set, and we can work with when we'll open the gate to discovery because that's, of course, a concern. If the pleadings aren't set, I would want to give you a trial date because I'm setting now in March of 2018. So I don't want to delay you beyond how I'm handling other cases, but reserving a hearing date on a motion that hasn't been filed is just not something I do. So I'm not inclined to do that.

But I do think it will -- it can save you a couple of months by getting the trial date at the earliest possible time, which is much longer period of time than you would want, but you're looking at a -- at least two years to trial once you have -- for your initial case management. It's been about 26, 27 months, is what I've been noting.

Now, in an antitrust case, that may not be that long, but you've been involved with each other for so long that these issues might not take as much to develop. I don't know.

I obviously hope that by having an early trial on the first two issues, that the -- that this -- that the tail -- the antitrust case may resolve through a recognition of how the issues have gone on these other things and you'll resolve the

whole case or maybe even resolve the whole case short of a trial, but nothing has given me hope for that.

UNIDENTIFIED SPEAKER: I understand. Thank you.

THE COURT: Okay.

UNIDENTIFIED SPEAKER: And I completely understand the Court's --

THE COURT: Yeah.

UNIDENTIFIED SPEAKER: It is a very important case for Arista, so we appreciate anything that can be do to keep it (inaudible).

THE COURT: Okay. Appreciate that.

So today, Mr. Pak, we'll certainly start with you. It's your technology, and are you presenting or is --

MR. PAK: I am presenting today.

THE COURT: Excellent.

MR. PAK: You have me for an hour.

THE COURT: Okay. That's wonderful. And I have your identification of the 10 claim terms. Thank you all for that. I appreciated that. And I'm -- I've read through the briefs, but of course I need to know more about the patent. So that's what we're here for today.

MR. PAK: So if I may hand up some stuff.

THE COURT: Yes. Thank you.

MR. PAK: So, Your Honor, I thought what we would do today is really start from the basics. Start with what kind of

hardware we are talking about, what is the software that runs on this type of hardware, get into some of the terminology that you will be hearing a lot about in this case.

Some of it will overlap the copyright just because we have CLI technology that is at issue from the copyright perspective as well as the patent perspective, but today's presentation is really focused on the patents. And once we have some of that foundation laid down, then we'll hit the '526 Patent first, and what I thought would be most helpful is to give you a sense of the motivation behind these patents as to the problems that were being addressed, the solution at a high level. Of course we'll talk about some of the examples of these solutions. We're not here to construe the claims today, so I'll try to keep my comments more factually focused on the technology rather than any kind of a legal argument on the claims construction.

THE COURT: Okay.

MR. PAK: Then we'll talk about the '886 Patent which is also a different set of problems and solutions related to (inaudible) technology.

Today is really for Your Honor's purpose and benefit so if there is anything that is unclear or Your Honor would like --

THE COURT: Good, thank you.

MR. PAK: -- (inaudible) just let me know.

So moving on to the presentation from Cisco, we have on

Slide 2 some pictures that I thought would be helpful to Your Honor to see some pictures of (inaudible).

THE COURT: Right.

MR. PAK: And they vary in all kinds of sizes, so when we talk about networking equipment, it could be literally the size of what we see here in terms of binders. It could be racks of equipment and it could be in places like data centers, think of Facebook, Google, Amazon. These are huge facilities that are being created with their own refrigerators, refrigeration systems, backup power centers and they could host thousands if not tens of thousands of different switches and servers.

All the way down to the smallest router and hubs that you might have at your home for wireless access. These are also a type of routers and switches in the sense that they create networks and allow you to correct with the internet or may be able to set up WiFi.

So the form factors may change, but the basic concept we will talk about will cut across all these different types of equipment.

So when we talk about networking equipment, we're talking about specialized devices that allow computer engineers and users to create networks.

In terms of terminology on page 3, you will hear a lot about switches and routers, and it turns out that there is a --

some disagreement in the industry about exactly what is a switch versus what is a router, and I've had a chance to depose some of the witnesses in this case about that, but at a high level, the general concept is the following.

A switch allows the networks to be formed in the first place. So if you have a set of computers -- if you took all the computers here in our room and printers and screens and we linked it up into a network, the device that would be used to create that network would be a switch.

Then -- and as we can see on the left-hand side of this presentation, we have a number of devices that are now commanded to the switches, the four networks.

And there is something called a router, and a router basically connects networks to other networks. So this is a bridge between network. So when you create what is called a local area network, which would be all the devices in the courthouse, and if one were to connect that to the internet or to other conference rooms in the building, you would use a router. And a router's job is to take traffic or packets from one network, translate it, if necessary, but place those packets onto another network and create this link.

So if you look at this screen here, the device in the middle that is connecting to the internet would be a router and then these would be the switches that would be connected by the router to the internet.

As I said, these terms sometimes get interchangeable, but that's basically the concept of switches and routers.

Now, the types of switches and routers that we will be discussing in this case are smart routers and smart switches. What do I mean by that? Just like we have smart phones, and if you think about what a smartphone is today, it's really a mini computer that's running inside of your phone. So instead of just making calls and receiving phone calls, it has the ability to run apps, applications. They have an operating system so if you have an Apple iPhone, it has an Apple operating system running on it. If you have Android device, it has Android operating system.

Similarly, these networking equipment from companies like Cisco and others are smart in the sense that they also have processors, they have a memory, and they have an operating systems. So when we talk about IOS -- and I think Your Honor may have seen that in some of the pleadings -- that's the Internetwork Operating System and that's a brand of an operating system that is manufactured by Cisco and marketed, just like Microsoft has Windows, which is their marketed version of their operating system.

And so the operating system is a software, it's -- sometimes we call it a platform in the sense that it's the basic level of software that you run on a piece of hardware to allow other applications and other software to run on the

device.

And so in Cisco's terminology, we call that IOS. Sometimes you will see NXOS, is another operating system variance from Cisco. Or sometimes we extend the IOS with abbreviations, IOS/XR or XE, and those are different flavors of the operating system.

But just like with Cisco IOS, now we call the people that actually interface with the network, system administrators or network administrators. Sometimes you will see the abbreviation *CIS admin*. These are people that actually administer the system.

THE COURT: These are actually human beings.

MR. PAK: Human beings. They are part of the IT departments of colleges, universities, companies, courthouses. And so these human beings, these system administrators, interact with the software that allows the switching equipment and routing equipment to be configured.

And so we'll get a lot more into this, but one of the things that system administrators want to do is configure the switches. So all these switches have lots of settings and parameters and features that can be activated, deactivated, different options turned on, turned off.

A network administrator assisting will send commands. One of the commands we will show here is *show interface*. And that's a command that this particular operating system can

recognize to say ah, the administrator wants to see all the interface that this particular switch can run, and that information is then relayed back to the system administrator.

So as I mentioned with the operating systems, one of the things that you would have to do is extend the functionality of the system by adding in applications and downloading them and installing additional applications that do further things than just the operating system. Just like you could download the Word application onto Windows or on Excel or on a smartphone, lots of different apps. You could do the same thing with a Cisco operating system.

And one of the types of tools we will talk about is something called Operations Administration and Monitoring Tools. OAM. And this is the topic of the '526 Patent, and if you read the patent, it will talk about these types of tools. And these types of tools allow a system administrator to monitor the performance of the system and look at all the different performance attributes of applications that are running, how fast are the switches running, is this application running properly, if there is an error message. All of that is monitored and configured with these Operations Administration and Monitoring Tools. And they run on top of the operating system. So (inaudible) like Microsoft Word would run on top of Windows.

One common way to access networking equipment is to do

graphical user interface just like Your Honor can now boot up Windows and have a screen with different types of windows and be able to drag Window menus down and select using the mouse, network administrators can do that as well. And there is some benefits to this technology in the sense that you can visualize what you're seeing. There may be icons or visual representations of switches and networks. In some ways, it can be intuitive to use a graphical user interface.

And what we're seeing on Slide 7 is an actual graphical user interface that Cisco offers. It's called SNMP tool, and that tool, as you can see, has different configuration settings. They're displayed to the user in a graphical sense. You can click on the main, for example, tab, drag it down just like you would in Microsoft Word, have different commands that in both of the --

THE COURT: Uh-huh.

MR. PAK: -- (inaudible).

But it turns out that there is another way to access these switch (inaudible), and that's through the Command Line Interface or CLI technology, and CLI technology has its own set of benefits and features that ultimately make it very suitable for use in a networking environment.

And a CLI is a Command Line Interface where you are literally typing in commands, so you don't see anything visual in terms of a graphical element. What you're doing is you have

a prompt. It's like if Your Honor may remember the MS-DOS back in the old day with Microsoft?

THE COURT: That was before my time -- that was before my computer time. Not my age, obviously.

MR. PAK: So there was a prompt, and you would type in commands into that interface. This is a very specialized form of Command Line Interface that's used for routers and switches. And here you are typing in commands in text and then the system would execute those commands and then provide information back to the user. So instead of dragging menus or clicking open windows, the user is typing commands in text form and then receiving text out the back from the system.

So I wanted to take the opportunity to help all of us visualize what this looks like, so what we see here on this screen is we have the system administrator sitting at a laptop, and you can see the blinking cursor. The hatch tag is the prompt. So that is just indicating to the system administrator I'm ready for a command. And it's blinking.

So then the system administrator would type in a command, so in this particular instance, he typed in MPLS forwarding-table. That is a particular command. It consists of two words: MPLS and forwarding-table. MPLS turns out to be one of the networking protocols. Forwarding-table is a -- a table that's inside the switch that is used to forward messages using this particular type of messaging protocol.

So this command actually would then show -- so the first word is *show*, so the network administrators is telling the system I want to see the forwarding-table for your MPLS protocol. And so now this command gets sent, as we saw, to the switch and the switch provides the information from that table back to the network administrator. And that is what we see --

THE COURT: Uh-huh.

MR. PAK: So you talk about screen input, screen output, this is what we're speaking of.

THE COURT: Okay.

MR. PAK: It turns out that every switch has its own set of commands that you -- that you want to program into the switch. And generally speaking -- and we'll talk a lot more about this in the (inaudible) of the case, but generally speaking, switch manufacturers develop their own CLIs. They have their own engineers who come up with different ways of expressing the CLI commands.

So (inaudible), for example, which is how many lines of text would I want to see on the screen. So it's terminal length in the context of Cisco. The terminal is the monitor. Length is the number of lines. And it turns out in Cisco's system, if you type *terminal length* and you give it a number, for example, 25 lines, that's the number of lines that you will see on the screen.

Juniper engineers came up with a different expression for

that same concept. They said *set CLI screen length*. And so if you type those commands into the Juniper system and put 25, it will do the same thing. But, again, different commands, same functionality, and each of these switches has programs that recognize these particular commands.

And so the basic point of the Slide 11 is that commands are not made by machines, they're made by human beings for human beings, so human engineers would sit around in conference rooms or sometimes they would write e-mails to collaborate. Other times it's an individual engineer who comes up with his best idea for why a particular command should look the way it does. And it's not a science. It's more of an art. We're relying on some engineering experience. They're also relying on their judgment and aesthetics about what they think the right command should look like. So they go through typically lots of different options for particular commands. So if you're just trying -- Cisco engineers could have looked at all kinds of different ways of saying set the line, number of lines on the screen.

The point here is that it's a human decision on the command expression, and generally speaking, most of these commands are intended to be used by human beings. So that will lead to some of the issues that are the subject of the patents in this case. So because this is a human interface in the sense that humans are coming up with commands to be used by

other humans, sometimes when you start to automate the process or try to use software tools with this human interface, there might be some challenges, and that's the subject of some of the patents in this case.

Slide 12 is a very basic point, but if I took one of the commands that the Cisco switch doesn't recognize because it happens to be a Juniper command, for example, it's not going to work. So a command line interface sometimes -- we'll talk about that as general technology of using a command line interface, but that doesn't mean that everybody has the same CLI interface. It's, generally speaking, distinctive, focused on particular manufacturer preference for of the different kinds of (inaudible).

So that basic question, how does the switch understand the command and enforce that particular command style? And it uses something called a CLI parser. And a parser is a software component or a set of components whose job it is to analyze incoming input in the form of potential commands. And then it uses a grammar, just like we would use grammar, to determine what this particular command expression means, but it also can validate the command expression so it can decide whether a command expression is a valid command expression or not.

It can also make suggestions, and we will see some examples of that. So if you happen to forget what comes next when I type in *show MPLS*, one of the things you can do with the

Cisco system is to ask for help and then the system will suggest here are the possible remaining entries to fill out that command and allow the user to then rely on the automated feature rather than pure memory --

THE COURT: It's actually not different than any kind of Google search or anything that --

MR. PAK: Right. We see that a lot today in the -- with Google. If you type in a Google search word, you'll see suggestions come up. Of course the CLI technology we're talking about goes back many decades now, so this originated in the -- in the late 1980s or early 1990s time frame. But today certainly most interfaces that allow the user to type in search words or other kinds of input has this ability to autocorrect or auto suggest.

THE COURT: Sort of like prompts.

MR. PAK: Exactly. But the important thing is that a CLI parser takes the input and puts it against the grammar of some type, some set of rules to say is this a valid command, is it not a valid command. If it's a valid command, get the command to the right place to be executed, and then provide the input back, the output back to the user and also perform some of these other helpful features like autocorrect and auto suggest.

So let's take an example, Slide 14, where the user typed in *show MPLS forwarding-table*, that same command, and then you

see that as it goes into a parser -- this doesn't actually show up on the screen, but I wanted Your Honor to understand that what happens inside the parser is it takes that *show MPLS forwarding-table*, and it recognizes each of those words. It says ah, I understand, this is a show command. This has a need; whereas, if somebody had typed a different word that it didn't recognize, it would tell the user internal improper command.

It also then recognizes okay, what do you want me to show you? And the next entry is MPLS. Ah, this relates to an MPLS industry standard protocol. So this particular command now has to do with that feature of the switch, and then the next entry is a forwarding table

Particular (inaudible) now has to do with that feature of the switch and then the next entry is a forwarding-table so what within that particular feature, do you want to be displayed (inaudible). So there's a hierarchy in the sense that we're looking at (inaudible) and the next level of the command syntax is what type of interface do you want to show. And then the last step in that hierarchy or level of hierarchy is what is the specific inclination within that standard or feature that you would like to be displayed onto the user screen.

So there is an inherent hierarchy that is often found within the CLI parsers, and you can see how engineers can

(inaudible) and say should the word *show* appear first or maybe we should have a different command syntax where maybe the interface should go first instead of typing in *show MPLS*, the user types in *MPLS show*. And that's a different way of thinking about commands. In the first instance --

THE COURT: Does that make any difference to the user?

MR. PAK: It does in the sense that the networking industry gets used to certain types of styles of CLS syntax. And so this then is not a science. It's more of of an art in the sense that you're trying to create (inaudible) that would be easier to use and maybe could be elegant in the minds of some engineers, but it is a real difference in terms of how the parser works. The fact that you start to show MPLS which means that in that system, the primary function is figuring out the action, is it a show, is it a set, is it a configure set, and then you tell it what the interface is versus another way of doing the command is by showing the MPLS. I'm telling it what it is that I want to control and then specifying the actions.

So there are different ways of organizing it. The same words may appear, but the fact that it's reversed in order or may have other elements embedded in it creates a new syntax and that's actually quite important to the parser technology.

So here in this case, we recognize the show MPLS forwarding-table and the parser sends the information back in text format that can be displayed on to the user's screen.

So as I mentioned, one of the things you can do is in the Cisco system, you can type *show question mark* and in this case, *show question mark* then returns back to the user a list of options. So here as you can see, show (inaudible) ethernet. Show cards.

So if you think about it, that might be another reason why you may want to start with a verb in this particular syntax, is that to be able to see the different kinds of interfaces. Or you could have another system where by starting with the interface, you can see all the different kinds of actions that (inaudible).

One more features that parsers can perform is autocomplete, so as you can imagine, network administrators are constantly typing away at these screens, so to help them save time, you can have the system recognize parts of commands. So here SH was typed and the system automatically recognizes that that is the show command. So instead of forcing the user to type S-H-O-W, the user is just relying on the fact that SH was typed in --

THE COURT: Uh-huh.

MR. PAK: Before we move on to the '526 Patent, is there --

THE COURT: No.

MR. PAK: Okay.

So the '526 Patent was filed in June of 2000, so if

Your Honor remembers, this was the first big heyday in the internet boom and there were lots of companies who were trying to revolutionize the world through internet technology. And at the time, one of the Holy Grails that people were trying to solve (inaudible) was something called unified messaging. And unified messaging is the concept that communications and messaging can come in all forms. So it could be an e-mail message, it could be a fax message, you could have a voicemail message, or even what's called voiceover IP, which is you could use the internet to make phone calls.

All of those are different kinds of messages, and before unified message technology, you would have to open up a separate application to view fax messages. You would have to open up another application to look at e-mail messages. Unified messaging was designed to integrate all of these different forms of messages into a standard format that the system can understand. And then the user can use a single application to view all kinds of messages and different (inaudible).

So that was the idea of unified messaging. And Cisco had a solution in 2000 that they developed for unified messaging and this was actually the beginning of the '526 Patent.

So the engineers that developed the '526 Patent were working in this unified messaging area and where can you see here on the screen is an actual historical document from

Cisco's files, and you can see that they were trying to either connect all kinds of devices, bringing in messages of different types into the internet network, and then have these applications, common applications process all these different text messages so that the user doesn't have to worry about the different application because he's using e-mail versus fax.

So the product that was developed as part of the patent activity was something called a U-1 and that was a marketing name that Cisco used to develop -- to describe its unified messaging product. And on page 20, we see one of the historical documents from Cisco's files that shows how unified messaging was incorporated into these large networks. And this is actually the IT department for one of the state agencies I think in Utah where you have all kinds of phones, they're IT phones, switches, routers, and the important thing here is that it was a Cisco unified messaging component that was added, and by adding this product into the mix, it allowed all these different types of messages to be integrated and then centrally treated and presented in a unified fashion back to the user.

So one of the problems that you have when you start dealing with lots of different types of messages and tools is how do you then administer and maintain all these different tools. So we talked about operations, administration, and monitoring tools, and those are some of the tools that were used for unified messaging. And, for example, if they had

three tools that ran on the system, Tool 1 could have a completely different command syntax than Tool 2 for the same functionality. So in this case, *at view* was a command that was recognized only by Tool 1, not by Tool 2 or Tool 3.

So if you typed *at view* into Tool 2, it would give you a validation error. So instead Tool 2 recognized something else called AceView as its command. So now you have two commands that basically perform the same function, one for Tool 1 and one for Tool 2.

THE COURT: I'm sorry this sounds so basic. I'm not really sure I know what a tool is, though.

MR. PAK: A tool -- sure. A tool is basically another name for an application --

THE COURT: Okay.

MR. PAK: -- for a network management program. So when I talk about tools, I'm talking about software tools. So these would be just like Microsoft Word or a spelling checking program on Windows. A tool here just simply means a software that is downloaded on to the operating system.

THE COURT: Okay. Thank you.

MR. PAK: So here we have separate tools, separate applications, and they have different commands. So now if you had the job of the system administrator that was maintaining this large network with dozens of tools or potentially hundreds of tools, then you had to learn command syntax of each of these

tools separately.

So every time the company decided to install a new tool, your job got that much harder because now you have to recognize and understand and memorize all these different commands.

THE COURT: Uh-huh.

MR. PAK: And what's confusing is that they are commands that do the same thing. So they were developed by different companies so they have different syntax, so you can see how confusing that might be.

There's also something called syntax, as we talked about before. It's not just the way the words are used, but it's the way the -- the sequence in which the way the words are typed in could be an issue. It could be additional characters that you have to type in or keystrokes to actually enter the command.

For example, in Slide 24, Tool 1, that *at view* command could be a single line command. So I type *at view dash G* and I type *enter* and it does the command.

For Tool 2, however, I'm going to need to do additional key strokes. It might be multiple line command syntax where I talk in *base view dash H* and turn then I have to press another key G to do the same thing. So it's basically a recipe that the administrator has to learn and diversify each tool.

So on one tool, I'm using one command line. Here I may have to use multiple command lines to get the same functionality. Again, adds to the complexity and potential

error.

So in Slide 25, one of the problems that the '526 Patent was designed to solve is simply time. So as the system administrator was trying to manage all these different systems, imagine having to create separate configuration files and having to log in to separate systems to invoke the same functionality over and over again, so if you wanted to reset a new tool with a new date, you are now in a Pacific time zone versus Eastern time zone, you had to log in separately and do all of that.

Obviously, this can lead to also mistakes, and this is on Slide 26. So if the user, the network administrator, forgot to use the right command or used a different command syntax, he could lead to errors. This was a -- beginning to be a big problem, especially in this context of now having open systems where you are putting in lots of different tools, so this is -- just like Microsoft Word is one of the many applications that you go into Windows, you have the ability to add lots of applications (inaudible).

So the Cisco engineers came up with their solution in the form of what's called UMCLI as the name that they gave to their embodiment of the invention. UMCLI just stands for Unified Messaging Command Line Interface. And so this was their generic command interface, and what's meant by *generic*, I'll talk more about today and in the hearing, but the meaning here,

it's an interface that works across lots of different tools. So instead of having a tool specific CLI as in Tool 1 would have its own CLI that recommends *at view* versus Tool 2 that would recognize *base view*, we have a generic interface that works across --

THE COURT: So that generic is then going to translate into the command that the tool recognizes.

MR. PAK: Exactly.

THE COURT: Uh-huh.

MR. PAK: So that's the basic concept. And there's the architecture behind it, there are the techniques behind it, but the basic concept is this creates a CLI interface that allows the system administrator to work with all kinds of tools. As new tools get added, you don't have to learn something new. You can map the generic command interface onto that new tool. The tool now works for you and you don't have to learn a new tool.

So on Slide 28, let's take an example out of the patent. There's a particular command that's described as an example *watch ACB globals*. ACB was a technology that Cisco had at the time that basically was conference calls so you can set up -- one of the applications would allow you to set up different kinds of conference calls through the internet, and this particular command *watch* -- it doesn't use the word *show*. I'm using the word *watch*. It says ACB. So I want to watch

something about these conference call applications, and globals was a counter. It was basically a counter that was maintained inside the system that kept track of how many conference calls --

THE COURT: Uh-huh.

MR. PAK: -- would be active at the time. So if the system administrator wanted to know how busy is the system, how many conference calls were going in, he would type in *watch ACB globals* into the unified message CLI, and that would go into the generic main interface. As Your Honor mentioned, one of the things that this CLI did is then it translated that *watch ACB globals* onto these tool specific things that we talked about before, *at view*, *base view*. The UMCLI would translate *watch ACB globals* with the right syntax, with the right kinds of input to be able to work with each of these different tools.

And so you have generic command that goes into a generic command interface and then the output of that would be tool-specific commands that goes on to the specific tools that are controlled by (inaudible).

So on Slide 33, the patent gives us very clear descriptions of what needs to be generic in the sense of generic hearings that is not dependent on the syntax of any tool. So that's one of the aspects of it. And the command is based on the relative function of the command rather than tool-specific syntax. All that means is the user now just

remembers I type in *watch*, I type in the sender or the protocol that I want to watch, and I type in the counter, the type of counter that I want to watch --

THE COURT: So it's really no different than adopting the language of one of the tools as the generic and then doing the translation.

MR. PAK: It is, except that what you need is the -- the heart of the invention is then the infrastructure --

THE COURT: No. I understand. But in terms of this generic --

MR. PAK: Right.

THE COURT: -- language, you could have just adopted one of the tools?

MR. PAK: Exactly. You could have taken one of the tools, so if you wanted to basically say I want to genericize one of the commands, then one option would have been let's take the commands of a particular tool I like, let's program the UMCLI acknowledging it in a way that allows me to map those commands onto the other tools. So it's not that the commands themselves are inherently generic. It's the programming of the parser and the programming of these transmitters that allow you to map your selection of generic (inaudible) onto to --

THE COURT: Well, and I -- was it also -- was there a problem of companies staying -- feeling they had to stay within one company of tools that -- because there was too much

training and too much mistake and this -- I mean, I assume it enhanced competition among the different developers.

MR. PAK: It did. It did enhance competition because what it allowed the developer community to do is create their own syntax and have their own CLI, but then the customers can use a tool like this, the UMCLI, where they can then have a set of translators, have a set of parsers --

THE COURT: Yeah.

MR. PAK: -- so that you may have a distinct CLI, but now this technology allows me, as the customer, to map whatever I want --

THE COURT: Without the downside of the training and the --

MR. PAK: That's right.

THE COURT: Okay.

MR. PAK: So this is a way of allowing different tools with different CLI syntax to all work together in the same customer environment, and so this was an important development in that sense.

And we have the testimony of one of the inventors, Jeffrey Wheeler, who explained that in 1999, he created this UMCLI tool to help better administrate the unified messaging product. And you can see that the tool is responsible for things such as starting and stopping software, setting different levels in terms of logging, monitoring the session activity, and other

key functionalities.

THE COURT: Uh-huh.

MR. PAK: This was the motivation behind the patent. And on page 35, he explains the difference between this technology versus what existed before. He says previously you had to use lots of different tools, lots of different CLIs, and with this, you have -- maybe execute five, six, seven, eight commands to accomplish one of these functions. So this was the motivation --

THE COURT: Uh-huh.

MR. PAK: -- behind this technology.

Let's get to the architecture on Slide 36. So again the invention isn't just this idea of let's have a generic (inaudible). This is a patent. We're talking about tangible software architecture and possibly a hardware that's involved. And in this particular embodiment, what we have -- this is Figure 1 of the '526 Patent that we've annotated here. You have the system administrator typing commands that are generic commands into the UMCLI parser. That parser would then map generic commands onto something called prescribed commands. So this would be an intermediate step. Before you get to the final issue command, you may have some intermediate descriptions of what that the command does. So generic command gets mapped on to a prescribed command.

And in particular environment, then you forward to that

different translators and the translators would then turn the prescribed commands to all the different sequencing and the syntax --

THE COURT: Yeah.

MR. PAK: So this is an architecture where you have parser. It's a program to recognize this is a generic command. Now I need to map it onto one or more of these prescribed commands.

THE COURT: So the parser actually modifies the command into different --

MR. PAK: Yes --

THE COURT: -- I'll call it language, but it's computer to computer at this point.

MR. PAK: Exactly.

THE COURT: Okay.

MR. PAK: So what's happening is the generic command comes in to the parser. The parser can decide this command should map onto one tool, because the user wants to control one tool, or it may map on to multiple tools because the user may want to control multiple tools with the same command, and then the translators then are responsible for taking each of these prescribed commands and doing the further translation necessary to make sure that --

THE COURT: Is there a translator assigned to each of the tools?

MR. PAK: Exactly.

THE COURT: So that's what this looks like?

MR. PAK: That's right.

THE COURT: And so the parser chooses which translator to send it to?

MR. PAK: That's right. You could send it to one or more of these translators and then the translators then issue --

THE COURT: So your parser is your triage nurse.

MR. PAK: That's right. Exactly, Your Honor. The parser is really -- the parser's job, just like the triage nurse, is to get the patient to the right place, get the command to the right place and make sure that the command is prepped in the right way so that the physician or the tool --

THE COURT: In the translator, this next step is actually what's modifying the generic or the -- I'll call it the prescribed command and is further reconfiguring it so it can be understood by the tool?

MR. PAK: Yes.

THE COURT: Okay.

MR. PAK: So you can think of it either as configuring or transforming it or just mapping it.

THE COURT: *Mapping* is the word you use.

MR. PAK: You're mapping it on to something else --

THE COURT: Uh-huh.

MR. PAK: -- like what the tool can understand.

THE COURT: Uh-huh.

MR. PAK: So Figure 2 is just an example of what -- how this could be done in a system like this. What -- I'm not going to spend a lot of time going into this in detail today, but this is a series of tables. On the left-hand side, you see a table at the top. You can see the mapped words to what are called tokens. Tokens could be words or it could be letters or it could be numbers or it could be any number of things or data elements. (Inaudible) a word is not to a particular token in this case. The connection is not to a token one.

And then on the right-hand side, you have a series of pointers, so you can see that you start at root which is at the top and it's highlighted in the sense that you're going down that tree structure and so as you go down the next level down, you're looking, for example, at the generic command was *watch TCP connections*. You recognize that ah, *watch* is the first word that I recognize here. So that's T equals 8. That's the yellow entry. And what does that tell me to do? There's an arrow, so I have to do a further check to see if there is another part to this command. And that arrow takes me to the blue box. That's the second layer in the hierarchy, and now I'm checking to see do I recognize TCP or is there some other protocol or a feature that's being specified. Ah, found TCP. Check. I don't stop there. I got to go one more level down

and then I'm looking at -- am I looking for the word *connections* and I find *connections* and then that gives me the corresponding command.

THE COURT: Uh-huh. Uh-huh.

MR. PAK: So this is, however, just one example. So there are lots of different ways of doing command parse in trees. You can use tables and pointers like the one that was just shown here. The patent makes it clear that the invention -- as Your Honor knows in patents, we talk about inventions and we talk about embodiments.

The patent is very clear that the invention is not limited to the disclosed embodiments, so you could have other ways -- as long as the -- claim language is satisfied.

THE COURT: Sure.

MR. PAK: -- you could have other ways of doing this. And so one basic example would be what if you just extended the idea of tables, so we have just one table in the last example. Well, what if we created a series of tables and linked them in a way that it had the same kind of hierarchy, and so this is just one example. There are lots of ways of doing this, but, for example, in this table to the right, you would see the same T8. That's the token that you're looking for for the first token.

Once you get a map, that pointer that you saw earlier, this one just gives you instructions. It says see Row 2. Ah,

that means I go to Row 2 and look for entries in Row 2. So in Row 2, I recognize TCP and it says see Row 5. So it's now asking me to skip Rows 3 and 4 and go to Row 5 and look for the next entry, and that's where connections can be found.

So by having a series of these message tables or hierarchy tables, you can do a parse tree list. It does the same thing, but it doesn't use pointers, so there are lots of different ways that you could perform that functionality.

So let me just pause here. So the '526 Patent originated in unified messaging. It's not limited to unified messaging. You can use it for other applications, but the basic problem you are trying to solve is I have lots of different tools, all of them have their own CLI syntax. How do I create an architecture that allows me to work with many tools but doesn't force the administrator to learn the syntax of every one of these commands? And so this is an architecture, a software architecture with hardware elements that allow the administrator to map generic commands onto multiple tools and --

THE COURT: So does the '526 deal with the addition of new tools and how it gets mapped?

MR. PAK: Yes.

THE COURT: Is that part of the '526 as well?

MR. PAK: It is part of the '526 as well. So what you would do, for example, in the '526 is -- this is why we the

dot, dot, dots there so the idea is the system could preship with, let's say, a set of three translators because the system would have three tools --

THE COURT: Uh-huh.

MR. PAK: -- that were part of the package that you got. But if the consumer or the company bought a new tool, then you could add a translator to that architecture so it would be Translator No. 4 that would correspond to Tool No. 4.

THE COURT: So the translator has the ability to take an unknown language from a tool and be the bridge between the generic and this hithertofore unidentified tool?

MR. PAK: Yes. That's the purpose of that translator, so --

THE COURT: Okay.

MR. PAK: The software would be code --

THE COURT: Yeah. But here, once they're in place -- so it actually can take an unlimited number and actually get it ready to receive the generic --

MR. PAK: That's right.

THE COURT: Okay.

MR. PAK: So this is part of the inventive aspect of the architecture because you have a parser --

THE COURT: Yeah.

MR. PAK: -- that's universal and that's the triage nurse, but you have an unlimited number of translators that

could be put into that architecture, so it's not fixed.

THE COURT: Okay.

MR. PAK: You will want to correspond between the parser and translator and tool. You could add more translators. You could add more tools. And then by changing the entries, that parsing mechanism that we saw, you could then recognize more commands if you wanted to. You could actually add more generic commands if you wanted to. You could add more tools into the mix.

THE COURT: So both -- so you'd have a new translator specially designated for the new tool?

MR. PAK: Correct.

THE COURT: But the parser has to be able to speak to the new translator, so there's some modification going --

MR. PAK: That's right.

THE COURT: And you call it mapping.

MR. PAK: The parser tree to recognize it -- the first part would be the same, the generic --

THE COURT: Right. Sure.

MR. PAK: But on the back half of it you would say ah, I want to take that generic command out and also map it on to this other prescribed command (inaudible) command --

THE COURT: So this patent really has to do with behind the human aspect. It's not about the human aspect.

MR. PAK: That's right.

THE COURT: Okay.

MR. PAK: Now, it enables something very interesting --

THE COURT: Sure. Sure.

MR. PAK: Because now the system administrator doesn't have to memorize all these different commands. But the invention is really behind the scenes.

THE COURT: Okay.

MR. PAK: (Inaudible) how the software works.

So then let's move on to the '886 Patent, which is the next patent. And this comes a little bit later in time so this is 2005. So we talked about 2000 or the first patent. And what happened -- what starts to happen in 2005 time period, Your Honor, is that we start to hear of things called clouds, and it's a very amorphous word and people have different meaning for it, but the idea is that data and applications no longer are limited to your desktop. So it may live somewhere in the (inaudible) in this cloud.

And the idea -- the reason why they use the cloud metaphor is that it's intended to convey to the user and the application developer that you don't have to worry about the details of how the network manages your data or the administrator. The cloud operator will take care of that for you.

So all you're doing is you're posting things on to the cloud and you are getting things off the cloud. But there is

an intentional separation between the management of the cloud and the development and management of the applications and data that use the cloud.

And so cloud becomes very popular for both consumer companies who we get to see, companies like Amazon and others start to use cloud technologies and Microsoft. But we also see companies internally creating their own internal clouds because they are becoming global enterprises that have lots of employees all around. They don't want to replicate the same database ten different times just because they're in ten different countries. They want to have a centralized network where they -- where anyone in the company can publish data and applications and others can use it.

So part of that cloud technology then, the challenge is when you deal with the CLI technology that's been developed, that was really intended for people to sit at computers and type commands. How do you make that work with the cloud architecture? And that's part the motivation for the '886 Patent.

So before we get to the motivation, I'll hit some of the foundational building blocks here. There's something called markup language, and in the computer science world, we talk about programming languages. So programming languages are languages that people write on machines to understand. So Java, C, C++, those are really humans writing code in a

particular language that the machine can ultimately understand. So that's a program language.

Markup language is something slightly different. Markup language is a way of annotating an existing document or a data to give it more meaning. And what do I mean by that?

So a common example here would be HTML, which is a hypertext markup language. This is really the backbone of the internet today. It was invented by a gentleman named Tim Berners-Lee at CERN in, I think, the early 1990s. His vision was to allow research scientists all around the world to publish their papers and share information.

But one of the things that researchers love to do is cross-reference other things, so they would write an article and say go see this article that talked about the problem before or here's my colleague's paper on this topic.

He created a system called hypertext where this is what we do every day when we use the internet. We click on links and it takes us to another document.

HTML was the markup language. It was the language that was used by the engineers at CERN to be able to create links. If you think about a link, we're not really changing the underlying content of the document. You're annotating the document at certain places to say this is a link, and we usually represent that by an underlining or some other holding and say to the user this is a special type of content. If you

click on this, it will take you somewhere. So that's a form of annotation. That's a form of marking up the document.

And so this is a little bit different, but in some ways, I suppose is similar to how, as lawyers, we mark up documents and we write things on the side and we say we want to take out this paragraph. Again, the role there is to edit the document. The role here is to give more meaning to actually allow the document to be more than just the content of stuff.

And so in this example, the HTML primary purpose was to take it to other links. HTML technology has evolved quite a bit. Now you can do lots of things. You can click on things and make it play video. You can actually run small programs (inaudible), but this was the beginning. So HTML is the classic example of a markup language.

And I don't know if Your Honor ever had this experience when you're on a website and you click on the wrong button in a browser, you might actually have seen something like this, which is the code behind the website page, and this is called the HTML document, and this is so really the code that actually is then processed by the browser to show you the wonderful, beautiful web page. But this is what HTML developed or would write. You would create a document where you type in all these different words and then make the document, but also he's telling the system I want this to mean something else. I want this to be a title. I want this to be a link. I want this to

be shown in a certain way.

So the thing that's used here is something called tag or tags, and a tag is -- usually it comes in pairs. So, for example, in Google, in this example, we want to say Google is a title. It's a type of a title. Google element is of the title so we want Google to be the title of this document.

To give that meaning to the document, we take this title tag and as -- the corresponding part of that tag is the backslash title, and they're both in these arrow brackets, so when the system sees ah, there is an arrow bracket and title, I know what follows next is a title type content and then where it ends is the other side of that tag, which is the backslash title. So everything that falls within that tag is now considered a title, so later on, you can create a rule that says, for example, anything that's a title, I want that to appear first. And I want that to be centered, and I want that to be in bold character. Those are all different types of things you can do in the makeup language.

Another part of the '886 Patent is something called an extensible. And extensible means instead of just using these tags that are predefined, for example, a title is something that comes as a basic part of HTML so every HTML document understands the task. Well, what if you wanted to create your own task. What if you, for your business or your research, use -- you want to have the ability to create your own

categories of documents and categories of content and data inside documents. That's why we have this concept of extensibility. So extensibility means that I can take the language and extend it to have my own tags. I can do other features. I can add other functionality to the markup language, and it turns out HTML itself is extensible and it's been extensible for a long, long time, including some of the very early versions where you could, instead of just using the title tag if you wanted, for example, create a tag that said *court exhibits*, so every time I type in this word or another word, I want to indicate to the system this is a court exhibit, I can create a court exhibit tag. So anything that falls within that tag, the system will understand that's a court exhibit. Or this is a deposition transcript, create a depo tag. This is a way of creating your own set of tags that's extensible.

So in Slide 43, we have a document that talks about HTML as an extensibility markup language -- extensible markup language where you can create something called classes, and classes just means categories of things, and you can create your own classes or categories to use HTML in a way that's extensible.

And that's important because the '886 Patent -- we'll talk more about this at the Markman hearing -- uses the phrase *extensible markup language* in the claims and throughout the

patent as well. And so we just have to be careful that when we talk about extensible markup language as that phrase, it can mean lots of different things. It can mean, for example, something -- we just saw there HTML extensible. It can be something actually abbreviated XML in all caps and that's a particular industry standard that uses a particular type of extensible markup language --

THE COURT: So XML, that's standard, but the phrase *extensible markup language* is the broad term.

MR. PAK: That's right. So XML is one form of extensible markup language. XML itself is an industry standard, and if you look at the industry standard, there are all kinds of variations of XML, depending on which year you look at, which versions of XML.

And then there's something called JSON, J-S-O-N, which is a javascript language, which is also another way of marking up documents or data and providing --

THE COURT: Uh-huh.

MR. PAK: But one of the things that we'll talk about is what does it mean to be extensible, what does it mean to be a markup, what does it mean to (inaudible) language, and then how does this all work together to create the solution in the '886 Patent.

So one of the things we can do, for example, if we're now going back to CLI, so if you were to run an automated CLI and

use this extensible markup language, instead of creating court specific tags, we might create CLI specific tags because we may be interested in understanding something as a CLI command.

So with the extensible markup language, you have the ability to create CLI specific tags that could be used by the system to recognize that this is a CLI command, this is a CLI input, this is a CLI output. These are specific to the CLI context, and you can create your own versions of these tags to deal with CLI technology.

So those are the building blocks. So let's think about the problem that the '886 Patent solves. This is 2005 again.

As we saw, CLI technology was developed by humans, for the most part, for other humans, and so here they're tightening in these messages or CLI commands on individual terminals and talking to individual switches.

Well, what if you wanted to control thousands of switches, if you want to have a single system administrator in Palo Alto communicate with a thousand switches distributed in the cloud or in the network somewhere, how do you make that happen? And you certainly don't want to fly him around to a thousand different places and have him log in to a thousand different computer screens and type a (inaudible) command. So this is where the concept of automation comes.

So automation means how do you automate or speed up the process of doing something manually. In Slide 46, we have in

the '886 Patent, Column 1, one of the problems that is specifically identified in the patent is that while a human user of IOS CLI -- and this is the -- IOS is the operating system that belongs to Cisco. CLI is the command line interface -- may be able to sort through the complicated input and output screen to create information and extract information (inaudible) data. It has proven to be a very difficult and cumbersome task to automate.

So what does that mean? It means that because different engineers over decades of time came up with their individual ideas and commands, and while there may be some overall stylistic guidelines, people were still selecting their own commands for other people when they got to the problem of trying to automate this process of recognizing this is a particular type of command and creating what are called scripts. Scripts are software tools that, for example, can run the same command on lots of machines, and so instead of the user typing in the same command, he runs a single software program, and basically that single software program on his behalf sends out those thousand commands to a thousand machines.

So scripts are basically -- sometimes we call it badge processing, but the idea is that you are automating a manual task by writing a software program that repeats the task over and over again. But every time it repeats that task, it will

do it on a different machine and it will do it with a different set of input, potentially. But the idea is the user, instead of typing in a thousand different commands, writes a program once and every time he wants to reset all of the switches around the world, he runs one program.

So it turns out the problem here was in this cloud world where there are lots of switches everywhere and (inaudible) how do you automate this process.

So the answer lies in using XML or extensible markup language or another version of extensible markup language to automate this process. And extensible markup language is beneficial in this regard because it's a language that both, if you created the right CLI parsers -- one thing you have to do is remember the CLI parser is the thing that understands. Now, if you go teach it how to understand extensible markup languages, so part of the invention here is to work with the existing CLI parser technology, to extend it to be able to work with extensible markup language so one of the languages now that it can understand is extensible markup languages.

But if you can modify the CLI parser or extend the CLI parser to provide support in this world, now you have languages that can be understood by both the computer you're typing on as well as each of the switcher (inaudible). That is a -- that's really a machine-to-machine language. This is a language now that is written for automation. It's written to automate

scripts and provide this type of programmatic interface to the parser.

And so if you could do this on both sides and you had to do it on both sides because the parser doesn't do any good if it doesn't understand that particular extensible markup language, but if you could do it on the parser and you could have it run on your desktop computer, now the user can type -- create this one markup language, extensible markup language script on the left-hand side, send it all around the world, across the network, and on the receiving ends, each of these switches will know how to execute that particular command.

So let's take an example of that. On Slide 48 here we have an example where you see these -- every time you see this arrow bracket, those are tags. So you have a --

THE COURT: Uh-huh.

MR. PAK: The next tag is a K underscore MPLS label. You recall that one of the commands that we showed was *show MPLS*. So here it's saying that this is a command, and it's a command that belongs to the MPLS category.

And then the next thing is K underscore range, so within that command, you have a range parameter or input into that command.

So a way to think about this is -- I forgot the name of it, but those Russian dolls that sat on top of each other and get smaller and smaller. Similar concept. So inside the very

middle, you will see first set of tabs that say okay, this is the middle K static, K underscore static. That tag has the back slash K static, so that piece of it is giving you the information for that piece of a command.

Once you unwrap that, you get to the next layer and the next layer and the next layer. So by creating these layers of encapsulated (inaudible), you can create a command structure inside of the markup language.

So what you do is I write the script. I say I want to now basically code my CLI command using the extensible markup language. This is just one of example. There is lots of extensible markup languages. Now I send it across the network. It ends up on the CLI parser that also understands that language. And on the switch, what happens is we start to parse that markup language, and so first thing is we recognize, ah, there's an MPLS label and this is an MPLS label command. So you can see there in the green box, there is a CLI, so it's embroidering or translating that markup language tag into an MPLS label. The next part of that command is range so grab that going further down into the middle of that set, and now I have the values for something called static and that was the middle piece and I now have my final piece.

So by taking this extensible markup language with the (inaudible) tags, by parsing it the right way, I can now have a CLI command on the other side. So what that means, I can then

use my (inaudible) CLI parser to just process that as any other CLI command.

So the beauty of this system is I'm not abandoning the CLI parser. I'm taking the CLI parser and I'm extending it by using extensible markup language, and so by packaging the CLI commands using an extensible markup language, I have now the ability to send this to the other side and now just so you know, you can't get the output back. So we have to do the other side of this equation.

So when you run the CLI command, the router will say command completed. It will give you some information to return to the user. We have to do the back side of that translation where here now on Slide 54, for example, the output responds to command completed. I need to communicate that back to the user on the other side of the (inaudible).

Here I'm going to (inaudible) now my extensible markup language with tags and I have created a specific tag, a CLI specific to show this example. Command completed is the response. I use my tags to create that. And then I send the extensible markup language across the (inaudible). And on the other side of what we just went through will happen. So they will unpackage it, recognize it as a command response and send it back to the user.

So to summarize again, this is also behind the scenes. This isn't about necessarily what the user sees, but it's

really about how you move CLI commands from one location to another or automate the entry of these commands across lots of systems, and the solution here is to use an extensible markup language, but use it as part of the CLI parser. The CLI parser is still doing the hard work of recognizing --

THE COURT: Where is this all built? This is built by humans?

MR. PAK: The --

THE COURT: The tags?

MR. PAK: The tags would be built by humans or it could be programmed. So it could be that the tool, another tool --

THE COURT: Uh-huh.

MR. PAK: -- could actually automatically generate the (inaudible) information. And that's part of the beauty of this. Is that instead of having the programmer or the user type right into the extensible markup language, they may be sitting at a tool that just has buttons that says here's the CLI command that I want to use. I'm going to select the network routers. I push one command. Behind the scenes, the computer then automatically generates the XML markup language representation of that command.

THE COURT: So this is done at a big level and it's somewhat static, I would presume, in the system.

MR. PAK: Yes. That's right.

THE COURT: So the program administrator, who's the person, they're not involved in this at all?

MR. PAK: They may not be involved at all. They may be behind the scenes.

THE COURT: Right. But this -- so the -- I don't know whether the company who owns this whole thing creates its own extensible markup language and has it uniquely created --

MR. PAK: You could do it that way.

THE COURT: Or you can get it -- you can have it off the shelf.

MR. PAK: Off the shelf.

THE COURT: Okay.

MR. PAK: But some of the system administrators may write their own XML or extensible markup language using these other programming languages that we talked about, JSON, but the basic concept is somewhere in the system, if you can enable the parser -- you have to change the parser to some degree.

THE COURT: Yeah.

MR. PAK: So the parser will understand this language, but you can have all the parsers understand this language and the desktops understand this language, then it allows --

THE COURT: Okay.

MR. PAK: -- the XML commands to be sent out --

THE COURT: Okay. Thank you.

MR. PAK: Thank you so much.

THE COURT: Thank you. Thank you. Are you ready for the other presentation?

UNIDENTIFIED SPEAKER: Yes, Your Honor.

THE COURT: All right. Good.

UNIDENTIFIED SPEAKER: May I approach with some copies of --

THE COURT: Yes. Thank you. Thank you. You know, we don't put these in the record itself. These are just -- and we're not reporting this or anything. This is just for my benefit today.

UNIDENTIFIED SPEAKER: Understood, Your Honor.

THE COURT: I don't want to hear any of this come back during trial.

MR. KRISHNAN: Good afternoon, Your Honor. Ajay Krishnan from Keker & Van Nest. I'm going to be presenting a short module at the beginning just about network basics.

THE COURT: Okay.

MR. KRISHNAN: And then I'm going to get into the '526 Patent and then my colleague, David Rosen, will be addressing the '886 Patent.

THE COURT: Okay. Sounds good.

MR. KRISHNAN: Some of this will possibly overlap a little bit --

THE COURT: Sure.

MR. KRISHNAN: -- with what Mr. Pak just said. I hate

going second --

THE COURT: I know. But it gives me a second chance as well, though.

MR. KRISHNAN: Okay.

So I want to start with just some very basic concepts about networks. Networks at a very simple level are different computers that can communicate with each other.

Why might you want to have a group of computers communicate with each other? There are all sorts of reasons that are probably just completely self-evident today. You might want to send an e-mail to somebody else. You might want to share files. You might all want to use the same printer. All of these are reasons why computers in a particular area may want to communicate with each other, and the earliest networks used something called a hub to do that.

So you see here a depiction of a hub. It's sitting in the middle and all of the computers are connected to the hub.

Now, the way a hub worked -- and, again, these are very rudimentary networks.

A user wanted to send a message to somebody else. They send a message to the hub, and what happens is the hub broadcasts it to everybody else on the network, whether they need it or not.

So that's a bit of a problem, but it does result in communication occurring. If you don't need the message, you

just ignore it, but that was the nature of hub.

The big advance came with switches. So the switch brings the advance of addressability. In contrast to a hub when the user wants to send a message, they send a message to the switch, just as they would with a hub, but now the switch will just send it only to the computer to which it was addressed. And so that -- that's really where switches come in. We'll be talking about switches a lot in this case. Switches are the main product that Arista makes. Cisco makes switches as well, so you'll be hearing about those a lot. You're not going to be hearing a lot about hubs.

Before I move on, I just want to introduce one piece of vocabulary. Mr. Pak used it very briefly. Local area network or LAN. You will hear it regularly with regard to networks. They are just a group of computers in a single geographic area.

So the example of all of the computers in this courthouse that are connected to the same network are part of a LAN. My law firm has a single office. All of the computers in the firm connect to the same work network, part of a LAN.

How do switches do this -- this addressability? They use something called MAC addresses. So MAC addresses have been around since the early 1970s and you can see at the top of the screen an example of what a MAC address looks like. It's a -- it's a -- this one is a 16-digit hexadecimal code and it is imprinted in the network cards of computers at the factory and

they never get changed. So they're in there for the life of the network card, and it's sort of like a social security number for a computer. And you can see here a picture of a network card that -- that's something that was just taken out of the computer. You can see the panel here, which you would see on the outside of the computer where you can plug in your ethernet for it. Inside the computer is this network card, the green thing, and somewhere in there its MAC address is implanted on it so that the network card has that and for the life of the network card it will have that MAC address.

And -- and that MAC address is what the switches use to address a message from one particular computer to another. They know the MAC addresses of the computers and so they can just send something to a particular computer as opposed to having to broadcast it widely.

And now we move on to routers. So routers are a different network device, and what routers do is they connect the different LANs to the internet. They connect routers to other routers, and it's this sort of system of architecture where you have local area networks that are connected by switches. Each of those is connected to a router. All of those routers connect to each other, and now you have your massive decentralized internet which is the amazing thing that we now have.

And one of the interesting things about this is that it is

completely decentralized. We have the cloud there, and it depicts the internet. It's not as if there is some magical computer in the center of it. It's all just routers --

THE COURT: Yeah.

MR. KRISHNAN: -- connected to each other.

Routers do this and they're now responsible for communicating all of the messages between these networks to other networks. They do not use the MAC address that we talked about earlier. Switches use the MAC address.

Routers use something called an IP address. Or Internet Protocol address. And these have been around since the early 1980s. IP just stands for Internet Protocol. That's one of the protocols that was developed, and the IP address is something that is issued pursuant to a protocol.

An example IP address is shown at the top of this screen here. It's that 10-digit number that you see. And the way that you're probably most familiar with some IP addresses at least is through web addresses. The web addresses are translated by their browser into the underlying IP address. And so instead of typing in www.google.com, you could at least at some point -- it actually typed in that number and it provides you with the exact same -- exact same website.

How does that all occur? It's through a central authority and that authority is called ICANN, Internet Corporation for Assigned Names and Numbers. It's a U.S. nonprofit that works

along with the U.S. government and they're responsible for allocating all of the IP addresses, making sure that two people don't have the same IP address because obviously that would result in all sorts of -- all sorts of miscommunication. (inaudible) story in the news this morning about an effort to try to move the power that ICANN has away from ICANN and to have it be more of an international body that governs it. But for now it's this U.S. nonprofit and the U.S. government that administers the whole system of IP addresses.

One of the interesting things about IP addresses is that back in 1981, when they were first developed, we were using 32-bit IP addresses and those are still actually used today. It resulted in 4.3 billion IP addresses being available. Come the mid 1990s, people realized that's actually way too few, and so they moved to something called IP V6, Internet Protocol Version 6, and that actually -- it allows for 128-bit IP addresses which resulted in a number of IP addresses that is so high I had to look up what it's called. The number is called 3 undecillion, U-N-D-E-C-I-L-L-I-O-N, and it's the number 3 with 38 zeroes behind it. So that's the number of IP addresses available --

THE COURT: It's like adding lanes on the freeway. It's already too small.

MR. KRISHNAN: Right.

So -- and this -- and this slide here just gives sort of a

rough concept of how routers use these IP addresses to route internet (inaudible). So imagine you have a computer in California that is connected to a router. It wants to access a website that's hosted by a computer in Maine. The router in California takes the IP address and it evaluates it. It uses a very complicated algorithm in the router to decide I need to send it along in that direction. There are thousands and thousands of routers it could send it to. It decides I'm going to send it somewhere over there because I can tell from the address it needs to go somewhere over there.

So that happens. It sends it to a router that's somewhere over there, which then a series of routers end up doing the same thing, using very sophisticated algorithms to --

THE COURT: Who manages these routers?

MR. KRISHNAN: I'm sorry?

THE COURT: Who manages these routers? The switch is inside the company, but whose routers are these?

MR. KRISHNAN: So, for instance, Comcast is a company that -- a lot of the internet companies and a lot of technology companies have massive routers that everyone uses. But it -- it's quite distributed and there are a series of different standards that make sure that everything actually talks on the same wavelength.

THE COURT: So they're not exclusive. So if I'm sending -- if I'm using Google, I'm not on Google's -- whatever

their network of routers is.

MR. KRISHNAN: That's --

THE COURT: Okay.

MR. KRISHNAN: And most people who make web addresses don't even control or have access to the underlying backbone. They're just --

THE COURT: Who maintains routers then?

MR. KRISHNAN: I'm sorry?

THE COURT: Who maintains routers then?

MR. KRISHNAN: The --

THE COURT: Each company that owns them?

MR. KRISHNAN: Yeah. Comcast or AT&T, they have massive networks and all sorts of other traffic --

THE COURT: Uh-huh.

MR. KRISHNAN: (Inaudible).

So this is sort of a -- the one thing I just want to say about this picture, it's simplifying things just a bit because these numbers don't precisely connect to like New England 63, Maine 142. That's a little bit of -- that's a little bit of a (inaudible) but basically --

THE COURT: Okay.

MR. KRISHNAN: As you get more specific --

THE COURT: Uh-huh.

MR. KRISHNAN: -- you can pinpoint -- (inaudible) in the right direction.

And finally I just want to go over the sort of spectrum of switches that are available commercially. And it is a very broad spectrum. On the -- on the left side of the screen, you see the most simple type of switch. This is something that could be used for -- for residential use. Oftentimes for residential use, switches and routers are combined into one single box so you don't have to buy the two of them separately. You can plug it in one thing that gets you onto the internet.

And then on the right side is sort of the bar -- other side of the spectrum. One of these massive managed switches that companies like Arista make that manage thousands and thousands of computers and massive quantities of internet traffic, they are managed, which means they need to be configured so we talked about this -- Mr. Pak mentioned these system administrators that need to be managing the switches. They have to -- they have to communicate with these switches and control them because there are all sorts of functionality you might want to add.

For instance, you might want to prioritize some traffic over other traffic. A switch can do that if it's managed. You might want to provide security to particular types of traffic. All of these things can be accomplished through a managed switch.

So now we'll move on to the -- to the '526 Patent. Here is the (inaudible) patent. It was applied for in June of 2000.

And the title is generic command interface for multiple executable routines and I'll get into what some of those things are in a minute, but I do want to address some of the background concepts first, the first one being the command interface or the command line interface.

So here you see an example of a command line interface. This was one -- this sounded earlier that Your Honor was not familiar with this from the 1980s. This is the MS-DOS prompt and the MS-DOS command line. So this was -- it was a pretty popular one in residential use in the 1980s. You can see the copyright here from 1981, 1982 --

THE COURT: It looks like my first computer though.

MR. KRISHNAN: Right. And it has the C there, the prompt. That's the command line prompt for entering textual messages.

And the way this worked is that a user would press -- type in their command and press enter in order to have that command be understood and executed by the computer. So the example I'll show here is the command print letter 1 dot text. The user types that in, presses enter, and then the computer prints up that page. And the -- the -- so the interesting thing for our purposes is well, how exactly did that happen? And it happened by use of something called a parser.

So the parser is a piece of software that receives the user input, interprets it and then sends -- sends on a command

to the rest of the computer to tell it exactly what to -- what to do, what the user wanted it to do.

So in this example, with print letter 1 dot text, once the user presses enter, the parser receives the -- the language that was typed in by the user so the word print, it gets the word print, it takes it one word at a time, it evaluates that word print, and it appreciates and understands that that is a valid command that it knows about. So it says okay, I understand, someone wants to use the print command. It's now looking for an argument to that print command, what do they want to print. It -- it then takes the next word, letter 1 dot text, and recognizes that as a file name. It has the dot TXT that indicates that it's a file. It recognizes it as a valid file and so it tells the rest of the computer I want you to -- I want you to print this document. So it issues a command to the rest of the computer that says print letter 1 dot text. And that's how these types of parsers work.

Parsers actually, by the way, are -- they're very fundamental technology. They've been around ever since textual input into computers has been around so at least the 1960s. And actually they're a common assignment for introductory computer courses, write your own parser.

THE COURT: Uh-huh.

MR. KRISHNAN: They're -- it's a commonly-assigned thing. I was -- I did it in high school.

So that was the example of a valid command. There's also invalid commands that -- that can be entered. Here it's -- the example PRINK instead of print. The user types that in and presses enter. What does the cursor do? It first takes that word print and it does not recognize it as a valid command. It -- there is no such command that returns the invalid command to the user. And similarly, if the user had typed the file name in incorrectly, dot TX instead of dot TXT, the parser would have recognized the print command, but when it got to the file, it would say well, I don't -- I don't know that file or that file type and return an invalid file name command to the -- to the -- to the user.

So of course any particular system will only be able to recognize those commands that it's programmed to recognize, but many different systems use the same command so, for instance, this print -- print file command was common among all operating systems at the time, and the same is true in the -- in the switch and router CLI context today. There are many commands that overlap between -- between many different (inaudible).

Another technology that is common to parser is this autocomplete functionality that we've talked about, and so this is a situation where the parser is not necessarily getting a full, complete, valid command from the user. There -- they are starting to type things in, and I think Your Honor noted that the -- it's the type of thing that happens in Google when you

start a search or e-mail programs. As you start the -- the parser is letter by letter instead of word by word looking at the letters that come in, comparing it to a list that it has of possible valid words, and you are giving the user a choice saying here are the possible valid words you could be using. Or if the user gets far enough along where it's -- this (inaudible), there is only one possible valid word that says --

THE COURT: Yeah.

MR. KRISHNAN: -- interrupt, I know this is what you want and I'll just complete it for you.

What I'm showing here is an excerpt from the 1980s. 1988 there was a operating system that was introduced by a company named DEC, Digital Electronics Corporation, that created an operating system for mainframe computers. It was called Tops 20, and here is an excerpt from the user guide describing the autocomplete functionality that it had, so to give a command using the abbreviated input, type only enough of the command to distinguish it from any other command. Usually typing the first three letters is sufficient to distinguish one command from another. Abbreviated input requires the least amount of typing of the various methods of --

THE COURT: Uh-huh.

MR. KRISHNAN: So this feature being able to sort of guess at what -- what the user is trying to -- trying to say is also a -- a feature of the parsers. It's something the parsers

can do.

The last sort of background concept that I want to talk about before getting into the patent is these OAM tools. Operating administration and monitoring tools. There are a couple of examples of these given in the patent. The patent isn't -- when you read the patent at the beginning, it's not terribly clear exactly what these are, but they give a couple of examples, and I'm trying to illustrate those here, what those are like. They give the example of something called realtime monitoring so realtime monitoring of the computer network. And they also give the example of an e-mail alert that is sent in response to a detective event.

So imagine that you have a system administrator, person on the left, and a network that they're monitoring, a number of people that are on their computers, and something goes wrong with the second computer. An alert is sent to the system administrator. That might be an example of an OAM or an RTM tool.

THE COURT: Uh-huh.

MR. KRISHNAN: So going on to the '526 Patent, this is the problem statement of the -- of the '526 Patent. It says, "The use of multiple RTM programs and other OAM tools, however, requires the users to remember the names and syntaxes of numerous commands for the respective RTM programs and OAM tools. Hence, an increase in the number of OAM tools would

result in the system administrator needing to develop expertise in the command names and syntaxes for the respective OAM tools."

And this is just a little bit of an extension of that problem statement and what the goal is of the invention. There is a need for an arrangement that integrates multiple document programs and command functionality for a user without the necessity of learning the respective command (inaudible) and syntax. And so here we get to the embodiment of the '526 Patent. This is how the parser that they describe works at a high level and I'll get into some of the specifics in more detail. But this is the high-level operation.

The user enters that generic command -- and I'll get into it in a second, to the generic command term. But the user enters it into a user input interface so the CLIs that we have been talking about, a keyword with the screen. That user interface sends it to the person just like we've seen, that piece of software, the first piece of software, and the parser of this patent uses in some embodiment to translation table, but in all events, this parse tree, and I'll explain how those work in a second.

And once that's done, a -- a command is sent from the parser through a specific translator to the particular downstream program that is supposed to receive the -- receive the ultimate command from the user. So that's -- that's the

high-level operation of the patented parser.

So this is generic. Again, it's a disputed claim term so I'm just going to state -- I'm going to read the sentence and then I'm not going to talk about it again.

It says, "As illustrated in Part A of the attached appendix, the new syntax provides a generic instruction that provides an abstraction of the tool-specific command, formats and syntax, enabling a user to issue a command based on the relative functions as opposed to the specific syntax or corresponding tool."

So that's the description it gives of a generic instruction set. To us it's not an (inaudible) clarity. The parties are going to dispute that, but I'll leave that for next week.

THE COURT: Okay.

MR. KRISHNAN: So this is Figure 2 of the patent, and this really now gets into the nuts and bolts of how this -- this parser works. It has a translation table up on the left. And then this parse tree down at the bottom. And just (inaudible), I'm going to go into a little bit of description of some of the elements of the parse tree, the first of which is called an element.

So each of these things in yellow is -- is describing a patent as an element. So at the top level of the tree, that thing that's labeled 24-A is an element. In the second level

of the tree, there are three elements. There is 24-B, 24-E, and 24-H. Each of those are elements.

On the -- on the third level of the tree down here, there are four different elements as you go across.

And then on this last level of the tree, there are two different elements, this one and that one.

So those are the elements. The next item are these token command key pairs. So within each element is one or more of these token command key pairs. So those are the things that are in green, and I'll get into -- get to tokens and command keys in just a second, but you see the T and the CK and those little boxes --

THE COURT: Uh-huh.

MR. KRISHNAN: The T is for token, the CK is for command key. So those are token command key pairs, and the first element there are three token command key pairs.

In this second element over here, there are two token command key pairs. And in all of the other elements, other than those two, there is actually only one token command key pair.

So -- and a command key -- I'll just explain it now, but you'll see how it works in a second. A command key is like a code that indicates what command will the parser give to a particular downstream management program. So the command key in a sense is the output of the parser. Once it knows what

command key it wants to -- it wants to execute, it can then send that to a translator that then sends the appropriate command to the downstream management program.

So that was actually all by means of orientation and now I'm going to give you a specific example. I'm going to use both of the examples that were in the -- that are just explained in the body of the patent. So these are the exact examples.

The first is *watch TCP connections*. This is a valid generic command. So the parser receives that input, watch TCP connections, and it takes it one word at a time. It takes the first word *watch*, goes to the translation table, and says watch the lines with Token No. 8. So now the parser goes down to the parse tree and it looks for a token command key pair in the top note that corresponds to Token No. 8 and it finds it. It -- that's the very first --

THE COURT: Oh, this is standard. This isn't built for each command?

MR. KRISHNAN: It is built for each command.

THE COURT: Okay. So when the user types in the generic command, this parse tree is created -- I'm not understanding when it's created.

MR. KRISHNAN: The parse tree is commanded -- is created before the user sits down and --

THE COURT: Okay. So this parse tree is always there?

MR. KRISHNAN: It has to be.

THE COURT: This is the template?

MR. KRISHNAN: Yes. And someone has to go in and type these -- it's -- I mean --

THE COURT: These are preexisting?

MR. KRISHNAN: They're preexisting and someone has to decide what the token command pairs are --

THE COURT: That's preexisting and somewhat static. It doesn't change all the time.

MR. KRISHNAN: Correct.

THE COURT: So when the command goes into the parser with the word *watch*, watch is assigned the token of 8.

MR. KRISHNAN: Yes. And it was assigned before anyone --

THE COURT: Right, right, right. Okay.

MR. KRISHNAN: So -- and by the way, you asked a question before about what if new management programs are added and you'll -- this will become clear in just a minute.

THE COURT: Okay.

MR. KRISHNAN: You have to redo the whole tree.

THE COURT: Got it. Okay.

MR. KRISHNAN: You have to redo the whole thing.

THE COURT: And so then once you have the token of 8, it is assigned the corresponding command key of 1.

MR. KRISHNAN: That has also already been done.

THE COURT: That's what I mean. Once you get to the 8, you're automatically paired with the 1.

MR. KRISHNAN: At this point in time. But the parser doesn't stop there because this is just the first word.

THE COURT: I understand. But just for the word *watch*.

MR. KRISHNAN: Exactly. Exactly. And that's -- that's exactly right.

THE COURT: Okay.

MR. KRISHNAN: So at this point, because it's found a token command key pair, the first note, it now knows that it is going to pay attention to the branch that's going off of that note. And something that I actually probably should have mentioned before is that if you look at each token command key pair in this tree, each of them has a particular branch that comes off of it. So those arrows --

THE COURT: Sure.

MR. KRISHNAN: -- are assigned to each one, each --

THE COURT: So what if -- if up above the word had been info, there's no *for* here, there is no token for *for*.

MR. KRISHNAN: That's right.

THE COURT: What would happen here?

MR. KRISHNAN: You'd get an error message.

THE COURT: That's an error message.

MR. KRISHNAN: Yes.

THE COURT: Okay.

MR. KRISHNAN: If you can't match the first word, you get an error message. If you can match the first word, then thereafter, you will never get an error message. You will see how this works in just a second. If you can't match the first word, you get an error message.

THE COURT: Okay.

MR. KRISHNAN: So the parser now knows because it got to that T8, it's now going to look down into the sub tree that is -- that that branch leads to and now it looks to the next word. The next word is TCP. The token of 6 --

THE COURT: Oh, I see.

MR. KRISHNAN: So now it's looking for a 6 in the sub tree that was -- that was -- that it was indicated by the --

THE COURT: So 4 is assigned to the sub tree. It just doesn't exist as a choice for the -- I understand now.

MR. KRISHNAN: Right.

THE COURT: Okay.

MR. KRISHNAN: If four is located somewhere in a sub tree, it means that info is not a valid first word.

THE COURT: Got it.

MR. KRISHNAN: So -- so we're now going down to this tree. It finds the word -- the token for TCP. It finds 4 -- sorry. 6. 6 -- in the token command key pair that's on the left of this element, so it finds -- it finds that token

command key pair and it now knows that if the next word is going to be a valid word, it's going to have to exist down in this element right here down here.

So the parser hasn't arrived yet at the end of the command so it goes to the next command. That's connections. It goes to the translation table. Finds the 2. And then -- and indeed the token command key pair for 2 is in there. So it will then say okay, I've reached the end of the command. It's a valid command. I've reached the end. Command key 3 is what the user is asking me to do. So -- or is what the designer of the system has decided the user wants when they type in *watch TCP actions* and therefore command key 3 will get translated into a particular command and sent down whatever actual command that is to a downstream management program.

THE COURT: So the parser then sends it to the translator at that point.

MR. KRISHNAN: Correct. I just want to make one little point here, is that some of the claims are actually a little ambiguous as to whether the translator is inside the parser or not inside the parser. So that's just an issue for the Court to be aware of. That it does -- some of the claims read as if the translation is happening inside the parser. That's not shown here, but it appears that some of the claims are --

THE COURT: Okay.

MR. KRISHNAN: So that's the valid command. And then I just want to walk through the example of the invalid command.

So this is -- this -- and this again is an example that is in the text of the patent. *Get UDP connection info*. We'll start with *get*. It's token 3. We look for the corresponding (inaudible) tree pair, token command key pair in the top element of the tree. We find it. We find that tree -- that 3, and now it wants to point its attention down to this element because that's the one that branches off from the token command key pair.

So it next looks for UDP. That's a 7. The token is 7, but the only token command key pair in that next (inaudible) is a 6. So it's going to say no dice. That's an invalid word. So what does this parser do? It goes back up to the last validated token, and in this case, it was that token T3, and so it says command key 6 is what I'm going to outline. So it interprets this command -- invalid command that starts with the valid word *get* as if it was just the word *get*.

THE COURT: That's going to give you nothing, though.

MR. KRISHNAN: Well, the designer has decided that command key 6 and at whatever command key 6 is, that is going to be (inaudible) of the command. So one thing I think you've recognized here is that this is a -- after you -- what you have -- you have to have a valid first word. If you don't have a --

THE COURT: Sure.

MR. KRISHNAN: -- a valid first word, you're not going to get into the tree at all. But once you have a valid first word, you'll never get an error command, error message back.

THE COURT: But it doesn't ensure that you're going to get what you want.

MR. KRISHNAN: Well, I think that what the -- I think -- probably the intention of the designer is to give you the thing that most likely people mean when they start --

THE COURT: Okay.

MR. KRISHNAN: -- with a word that -- that is --

THE COURT: Okay.

MR. KRISHNAN: But I have to make a choice. Every single token in this tree is matched to the corresponding command key.

THE COURT: Okay.

MR. KRISHNAN: So there's always going to be --

THE COURT: Sure.

MR. KRISHNAN: -- some -- something that -- that can happen.

And -- and this is what -- I was going to make the point, I think in response to your question, what happens if you add a new management program to the system. The -- the -- the designer would have to go back and redo the whole translation table and the parse tree to -- to accommodate that new

management program.

THE COURT: Uh-huh.

MR. KRISHNAN: And the last thing I'm going to talk about is this algorithm in Figure 3. This is the exact same algorithm that we just talked about except in a flow chart format, instead of showing what sort of (inaudible).

THE COURT: Okay.

MR. KRISHNAN: So it starts you parse the first word of the generic command by the matching token. That's going to the translation table. You traverse the command tree. Do you find the valid command word. And, remember, this is the very first word.

THE COURT: Uh-huh. Uh-huh.

MR. KRISHNAN: If you don't find it, you do this --

THE COURT: Yeah.

MR. KRISHNAN: -- invalid command error message. And the interesting thing is nowhere else in this --

THE COURT: Uh-huh, uh-huh.

MR. KRISHNAN: -- flow chart can you ever get to an invalid command error message except if you fail to match the first word. But if the first word matches, you're now into the rest of the flow chart. You just keep parsing the next word by the next word. So you traverse the relevant portion of the tree and each -- for each subsequent word, you ask is there a matching token command key for that. If so, do I stop here or

are there further words to the command. If so, I just keep repeating this process going down the tree --

THE COURT: Uh-huh, yeah.

MR. KRISHNAN: -- until I get to the last word. However, if any subsequent word after the first word does not match the token command key pair, then it just skips to the bottom there. It says map the command using the command key of the last valid command word.

So if the fourth word is invalid, it uses the command key associated with the third -- third word. If the second word is invalid, it uses the command key --

THE COURT: Yeah.

MR. KRISHNAN: So that is basically all I have to say about the '526 Patent.

THE COURT: Okay.

MR. KRISHNAN: And now Mr. Rosen will talk about --

THE COURT: Thank you.

MR. ROSEN: Good afternoon, Your Honor. My name is David Rosen, and I will be talking about the '886 Patent.

The '886 Patent is called the Method and System of Receiving and Translating TLI Command Data within a Routing System. So what problem is that trying to solve?

The Office of the Patent tells us right away CLI is a comprehensive interface which has expanded continuously as technology has improved over the past 20 years. Many companies

now strive to support some variation of IOS CLI in the routing systems. And many consumers have invested heavily in IOS CLI support, developing complicated scripts to handle various configuration and access needs. As such, it is desirable for any improvements to router access and control to acknowledge existing investments of consumers.

So as Mr. Pak said earlier, IOS is a trade name. It refers to Cisco's IOS product. We're talking about a system that has been around since the mid 1980s. We have -- the patent was filed in 2005 and then 20 years of development. And we have many consumers, customers of this IOS CLI product that have invested heavily in it, that depend upon it, and they are writing scripts which, as Mr. Pak said earlier, are just computer programs that interface with the IOS CLI and even include the IOS CLI commands.

The patent goes on. IOS CLI is not the most program-friendly of interfaces, however. Twenty years of consistency and backwards compatibility, coupled with consistent improvements to the hardware and implementation of new features has created an extensive interface.

While human user of IOS CLI may be able to scroll through the complicated and input and output scheme, input information, extract important data, it is proven to be very difficult and (inaudible) the auditing. An assisted (inaudible) method that allows for an easy, more structured approach to accessing a

(inaudible) router, while still making use of the (inaudible) advances and experience associated with IOS CLI, would be advantageous.

So, again, we have 20 years of development, we have consumers that are looking for an easier way to interface with the system, and Cisco to solve that problem is offering a more -- through this patent, it's describing a more structured approach.

And what form does that structured approach take? The patent describes using XML to take commands and send those commands over the internet or through the cloud, as Mr. Pak said earlier, to the CLI parser and then output X amount back to the consumer or the client.

XML is -- there's a formal specification for XML. It's called (inaudible) worldwide web (inaudible). The W3C. The XML core working group publishes the formal specification for the extensible markup language and maintains errata for that document.

Earlier in response to Your Honor's question, Mr. Pak said that extensible markup language refers more generically to different types of languages or technologies, and I just wanted to pass on a point of dispute in claims construction, we think it's clear that extensible markup language refers to formal specification of (inaudible).

THE COURT: Okay. We'll talk about that the next

time.

MR. ROSEN: Indeed. I won't belabor it.

THE COURT: Okay.

MR. ROSEN: And when talking about XML and describing it today, I'm going to occasionally refer to W3C definitions and descriptions of the XML language.

So one might (inaudible) we use XML. This is not XML. This is just text. It should look familiar to Your Honor.

THE COURT: Uh-huh.

MR. ROSEN: You will find it on the Northern District of California website. And when we look at this text, we know what it means as humans based on our knowledge and experience, and we know that Edward J. Davila is the name of the district court judge. We know that first street, San Jose, refers to a street address, and we know that because we have our experience.

But to a computer program or a computer application, this is just arbitrary text. I'm using our formula and this may as well be baseball scores on the PowerPoint. There is nothing here to tell PowerPoint what this means.

But sometimes we may want to use a programmer application that needs to define individual elements of text and that (inaudible) something. So one very bold example Your Honor may be familiar with is a mail merge. If we wanted to put this data into a mail merge, we have to define the elements. We

have to define the name, we have to define maybe the courtroom so that the mail merge could put the information in text on top of letterhead or (inaudible).

So XML is a way these mark up with tags to define discrete elements. Let's see how this might look in XML. It's the same text that we saw before, but now it's surrounded by tags that define it so that the computer program can make sense of it. So it can separate the name of the judge from the courtroom in which the judge sits.

THE COURT: Uh-huh.

MR. ROSEN: Each XML document contains one or more elements, the boundaries of which are delimited by start tags and end tags. So the start tag, starting with name, is the beginning of the name element. The end tag is the end of the name element, and then Your Honor's name is between this part of that element.

We also have -- we can have parent elements, we can have broader elements. This is the judge element. And the judge element starts at the start tag, it ends at the end tag. Everything in between this part is the judge element. You can see that the (inaudible) has sub elements like *name* that we talked about, like *courthouse*.

This is an XML schema, and Your Honor doesn't need to be too concerned with this. I know it looks complicated. XML schema is a term that occurs frequently in (inaudible)

specification, and I thought it might be useful to see what a schema looks like. An application doesn't have to use a schema. It can use a schema to describe the elements of an XML document and define them.

So just, for example, we see here that the name (inaudible) as a type of strain. All of these fields are strain, but you could imagine data that may be included in date, data type or a number data type and schema may be a way of defining those XML elements or an application that used XML.

So how does this look in the context of the '886 Patent? This is Table 1 and Table 2 from the specification. So Table 1 is the XML that would come in to the IOS CLI parts center. And Table 2 is the command that comes out after the parser has done its work.

THE COURT: This is in the patent?

MR. ROSEN: It is.

THE COURT: And this is right out of what you will talk about next week is the standard XML?

MR. ROSEN: Yes. We -- this is standard XML as an example. The plaintiffs disagree about whether --

THE COURT: It's limited to standard. Okay.

MR. ROSEN: This is a discussion of what might happen at the remote device, at the consumer's device, remote from the IOS CLI. If a consumer wants to send this command to the IOS CLI, it's going to need to format that command in XML. And so

we see on the right, you know, an add element. We have different elements and then different tags within those elements that contain command keywords like label and range and parameters, parameter values like 10 (inaudible).

Now, once the XML is formed, once the client application or the remote application, remote device has formed that XML, it's going to send it over the internet. And it will go to the parser.

So now we're at the parser. Now the XML has made it over to the CLI parser, and what happens? It's -- it's the reverse of what had to happen before. The parser now has to parse this command out of those XML elements out of those texts.

And then finally after the command has been parsed, it gets executed and then it's output. If it has any output, that output is going to be returned back to the remote device. Back to the consumer.

THE COURT: Right. Uh-huh.

MR. ROSEN: And that's the '886 Patent.

THE COURT: Uh-huh. All right.

MR. ROSEN: If Your Honor has any more questions I'll --

THE COURT: No. Thank you. Well, that's very helpful and that will give me some context for reading your briefs on the disputed terms.

All right. And that was -- I thank you for making this

exactly the time that you had suggested. That's really quite remarkable.

The last thing I need to do is to pick a date, and do you have the March and April calendar that you put together? Not the -- you know, the one that you do?

THE CLERK: I can print it out.

THE COURT: Just those two months, March and April. That's what I meant to bring. So we were set for the 18th and I'm either -- well, I'm probably going to put this off just two weeks. It's not -- it's really just a very short change.

What is the time estimate for the claims construction?

UNIDENTIFIED SPEAKER: Your Honor, I think the parties requested four hours so it would --

THE COURT: Oh, four hours?

UNIDENTIFIED SPEAKER: We should be able to squeeze it in three hours.

THE COURT: It's just that it makes it go into the afternoon, which makes it a whole different experience. All right. The first I have -- oh, all right. That didn't work out so well.

I think that I'm going to bump somebody else. What I'm thinking is that the best day for me will be April 1st. And then we'll move the tutorial for the other one.

UNIDENTIFIED SPEAKER: (Inaudible). We'll make whatever works that needs to work. Your question was just

saying that's the one day.

UNIDENTIFIED SPEAKER: (Inaudible) possible either --

THE COURT: I can do it on the 8th. I was just trying to give you the earliest day I was available after my trial ends and that actually then I don't have to move anybody. So is that better?

UNIDENTIFIED SPEAKER: I would appreciate it, Your Honor.

THE COURT: Of course.

Mr. Pak, how does that look for you?

MR. PAK: Your Honor, I think we can make that work.

THE COURT: I appreciate that. Thank you. And I'm sorry that this other trial -- all right.

UNIDENTIFIED SPEAKER: Should we do it in the afternoon?

THE COURT: No. Not if you want four hours. Actually, on that day, I need to be able to leave by -- I actually need to be out the door at 1:30. So I'm going to suggest that we start it at 8:30.

UNIDENTIFIED SPEAKER: Sure.

THE COURT: And my hope would be that we will finish without a lunch break. And then -- and so we can finish it. I'm going to -- obviously I slow you down with questions, but I'm going to really ask you to keep it to the time you've asked for and our goal will be to finish at 12:30. All right. I

think that works. Because I was unavailable for the afternoon there. Yeah. All right. I think that makes sense.

Yeah. That works. And I have your briefing and I -- I'm a little concerned about the rapid pace that this case is taking, and I guess we have one day for summary judgment motions for the whole -- both aspects of the case. And I just need to make clear to you that I am a stickler for you complying with the rules on brief length and I actually can't see print that's less than 12-point type. I will not read it. I'm not going to strike it. I will simply ignore it. You just have to understand I can't do it.

And I don't take exhibits that expand your -- so I -- I actually, on summary judgment, do not want a separate statement. I have that -- and it is causing lawyers to give me 30 pages of further argument, and so I want you to know that -- you'll be glad to know you don't have to do it. I know. But you will really have 25 pages for briefing on everything, and, you know, in my view, summary judgment is not a trial balloon to test out your opponent's arguments. So please choose wisely on issues that you think you validly have a chance at summary adjudication. I'm sure you've had that lecture many times. But I think that's -- I think you have your date for summary judgment. That should be fine. I'll get this claims construction out. If you're not back until April 8th, obviously it's going to put it off. It may dovetail pretty

close to the time you hear about the IPR anyway, but I'm not going to be concerned about that.

Okay. And so I'm hoping we can get some work done in advance on this so that the claims construction hearing is really more to answer our questions and I can get this to you as aspirational, I suppose.

Is there anything more that we need to discuss today?

UNIDENTIFIED SPEAKER: No, Your Honor.

THE COURT: It's been very helpful. Thank you all.

(Proceedings adjourned at 3:40 p.m.)

CERTIFICATE OF REPORTER

I certify that the foregoing is a true and correct transcript, to the best of my ability, of the above pages of the official electronic sound recording provided to me by the U. S. District Court, Northern District of California, of the proceedings taken on the date and time previously stated in the above matter.

I further certify that I am neither counsel for, related to, nor employed by any of the parties to the action in which this hearing was taken; and, further, that I am not financially nor otherwise interested in the outcome of the action.

Pamela A. Batalo

Pamela A. Batalo